Cambridge

OL IGCSE

Computer science

CODE: (0478)

Chapter 01 and chapter 02

Data representation and Data transmission

1.1 Number systems

1.1.1 Binary represents data

This book explains the complexity of computer systems and the software controlling them. It explains that data must be converted into a binary format for processing. The binary number system is the basic building block in all computers, representing millions of tiny switches in the ON or OFF position.

1.1.2 Binary, denary and hexadecimal systems

The binary system We are all familiar with the denary number system which counts in multiples of 10. This gives us the well-known headings of units, 10s, 100s, 1000s, and so on:

(10^4)	(10^3)	(10^2)	(10^1)	(10^0)
10 000	**1000**	**100**	**10**	**1**
2	5	1	7	7

The binary number system is a base 2 number system. It is based on the number 2. Thus, only the two 'values' 0 and 1 can be used in this system to represent all values.

The typical headings for a binary number with eight digits would be:

(2^7)	(2^6)	(2^5)	(2^4)	(2^3)	(2^2)	(2^1)	(2^0)
128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
1	1	1	0	1	1	1	0

A typical binary number would be: 11101110.

Converting from binary to denary

The conversion from binary to denary is a relatively straightforward process. Each time a 1-value appears in a binary number column, the column value (heading) is added to a total This is best shown by examples.

Example 1

Convert the binary number, 11101110, into a denary number.

128	64	32	16	8	4	2	1
1	1	1	0	1	1	1	0

The equivalent denary number is 128 + 64 + 32 + 8 + 4 + 2 = 238

Example 2

Convert the following binary number, 011110001011, into a denary number.

2048	1024	512	256	128	64	32	16	8	4	2	1
0	1	1	1	1	0	0	0	1	0	1	1

The equivalent denary number is 1024 + 512 + 256 + 128 + 8 + 2 + 1 = 1931

Converting from denary to binary

The conversion from denary numbers to binary numbers can be done in two different ways. The first method involves successive subtraction of powers of 2 (that is, 128, 64, 32, 16, and so on); whilst the second method involves successive division by 2 until the value "0" is reached. This is best shown by two examples:

Example 1

Consider the conversion of the denary number, 142, into binary:

Method 1

The denary number 142 is made up of 128 + 8 + 4 + 2 (that is, 142 – 128 = 14; 14 – 8 = 6; 6 – 4 = 2; 2 – 2 = 0; in each stage, subtract the largest possible power of 2 and keep doing this until the value 0 is reached. This will give us the following 8-bit binary number:

128	64	32	16	8	4	2	1
1	0	0	0	1	1	1	0

Method 2

This method involves successive division by 2. Start with the denary number, 142, and divide it by 2. Write the result of the division including the remainder (even if it is 0) under the 142 (that is, 142 ÷ 2 = 71 remainder 0); then divide again by 2 (that is, 71 ÷ 2 = 35 remainder 1) and keep dividing until the result is zero. Finally write down all the remainders in reverse order:

2	142				read the remainders from bottom to top to get the binary number:
2	71	remainder:	0		1 0 0 0 1 1 1 0
2	35	remainder:	1		
2	17	remainder:	1		
2	8	remainder:	1		
2	4	remainder:	0		
2	2	remainder:	0		
2	1	remainder:	0		
	0	remainder:	1		

▲ **Figure 1.1**

We end up with an 8-bit binary number which is the same as that found by Method 1.

The hexadecimal system

The hexadecimal number system is very closely related to the binary system. Hexadecimal (sometimes referred to as simply 'hex') is a base 16 system and therefore needs to use 16 different 'digits' to represent each value.

The typical headings for a hexadecimal number with five digits would be:

(16^4)	(16^3)	(16^2)	(16^1)	(16^0)
65 536	**4096**	**256**	**16**	**1**
2	1	F	3	A

The following table summarises the link between binary, hexadecimal and denary:

▼ Table 1.1

Binary value	Hexadecimal value	Denary value
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Converting from binary to hexadecimal and from hexadecimal to binary

Converting from binary to hexadecimal is a fairly easy process. Starting from the right and moving left, split the binary number into groups of 4 bits. If the last group has less than 4 bits, then simply fill in with 0s from the left. Take each group of 4 bits and convert it into the equivalent hexadecimal digit using Table 1.1. Look at the following two examples to see how this works.

Example 1

101111100001

First split this up into groups of 4 bits:

1011 1110 0001

Then, using Table 1.1, find the equivalent hexadecimal digits:

B E 1

Example 2

100001111111101

First split this up into groups of 4 bits:

10 0001 1111 1101

The left group only contains 2 bits, so add in two 0s:

0010 0001 1111 1101

Now use Table 1.1 to find the equivalent hexadecimal digits:

2 1 F D

Converting from hexadecimal to denary and from denary to hexadecimal

To **convert hexadecimal numbers into denary** involves the value headings of each hexadecimal digit; that is, 4096, 256, 16 and 1.

Take each of the hexadecimal digits and multiply it by the heading values. Add all the resultant totals together to give the denary number. Remember that the hex digits A → F need to be first converted to the values 10 → 15 before carrying out the multiplication. This is best shown by two examples:

Example 1

Convert the hexadecimal number, 4 5 A, into denary.

First of all we have to multiply each hex digit by its heading value:

256	16	1	
4	5	A	
(4 × 256 = 1024)	(5 × 16 = 80)	(10 × 1 = 10)	(NOTE: A = 10)

Then we have to add the three totals together (1024 + 80 + 10) to give the denary number:

1 1 1 4

Example 2

Convert the hexadecimal number, C 8 F, into denary.

First of all we have to multiply each hex digit by its heading value:

256	16	1	
C	8	F	
(12 × 256 = 3072)	(8 × 16 = 128)	(15 × 1 = 15)	(NOTE: C = 12, F = 15)

Then we have to add the three totals together (3072 + 128 + 15) to give the denary number:

3 2 1 5

To **convert from denary to hexadecimal** involves successive division by 16 until the value “0” is reached.

1.1.3 Use of the hexadecimal system

The computer scientists can work with binary, they find hexadecimal to be more convenient to use. This is because one hex digit represents four binary digits. A complex binary number, such as 1101001010101111 can be written in hex as D2AF. The hex number is far easier for humans to remember, copy and work with. This section reviews four uses of the hexadecimal system.

» Error codes
» MAC addresses
» IPv6 addresses
» HTML colour codes

Error codes

Error codes are often shown as hexadecimal values. These numbers refer to the memory location of the error and are usually automatically generated by the computer. The programmer needs to know how to interpret the hexadecimal error codes.

Media Access Control (MAC) addresses

Media Access Control (MAC) address refers to a number which uniquely identifies a device on a network. The MAC address refers to the network interface card (NIC) which is part of the device. The MAC address is rarely changed so that a particular device can always be identified no matter where it is.

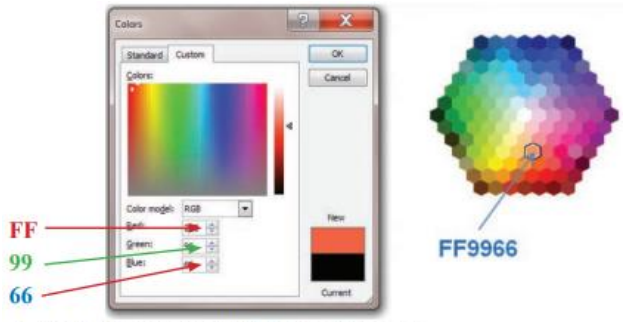
Internet Protocol (IP) addresses

Each device connected to a network is given an address known as the **Internet Protocol (IP) address**. An IPv4 address is a 32-bit number written in denary or hexadecimal form: e.g. 109.108.158.1 (or 77.76.9e.01 in hex). IPv4 has recently been improved upon by the adoption of IPv6. An IPv6 address is a 128-bit number broken down into 16-bit chunks, represented by a hexadecimal number.

Hypertext Mark-up Language (HTML) colour codes

Hypertext Mark-up Language (HTML) is used when writing and developing web pages. HTML isn't a programming language but is simply a mark-up language. A mark-up language is used in the processing, definition and presentation of text (for example, specifying the colour of the text).

The following diagrams show the various colours that can be selected by altering the hex 'intensity' of red, green and blue primary colours. The colour '**FF9966'** has been chosen as an example:



▲ **Figure 1.5** Examples of HTML hex colour codes

1.1.4 Addition of binary numbers

This section will look at the addition of two 8-bit positive binary numbers.
Note the following key facts when carrying out addition of two binary digits.

binary addition	carry	sum
0+0	0	0
0+1	0	1
1+0	0	1
1+1	1	0

This can then be extended to consider the addition of three binary digits:

binary digit	carry	sum
0+0+0	0	0
0+0+1	0	1
0+1+0	0	1
0+1+1	1	0
1+0+0	0	1
1+0+1	1	0
1+1+0	1	0
1+1+1	1	1

For comparison: if we add 7 and 9 in denary the result is: carry = 1 and sum = 6; if we add 7, 9 and 8 the result is: carry = 2 and sum = 4, and so on.

Overflow

The maximum denary value of an 8-bit binary number is 255 (which is 28 – 1). The generation of a 9th bit is a clear indication that the sum has exceeded this value. This is known as an overflow error and in this case is an

indication that a number is too big to be stored in the computer using 8 bits.

Example 3

Add 0 1 1 0 1 1 1 0 and 1 1 0 1 1 1 1 0 (using 8 bits)

```
  0 1 1 0 1 1 1 0
+ 1 1 0 1 1 1 1 0
  ---------------
1 1 1 1 1 1 1      <- carry
1 0 1 0 0 1 1 0 0  <- sum
```

9th bit

This addition has generated a 9th bit. The 8 bits of the answer are 0 1 0 0 1 1 0 0 – this gives the denary value (64 + 8 + 4) of 76 which is incorrect because the denary value of the addition is 110 + 222 = 332.

1.1.5 Logical binary shifts

Computers can carry out a logical shift on a sequence of binary numbers. The logical shift means moving the binary number to the **left** or to the **right**. Each shift left is equivalent to **multiplying** the binary number by 2 and each shift **right** is equivalent to **dividing** the binary number by 2.

1.1.6 Two's complement (binary numbers)

Up until now, we have assumed all binary numbers are positive integers. To allow the possibility of representing negative integers we make use of two's complement.

Writing positive binary numbers in two's complement format

Example 1

The following two examples show how we can write the following positive binary numbers in the two's complement format 19 and 4:

−128	64	32	16	8	4	2	1
0	0	0	1	0	0	1	1
0	0	0	0	0	1	0	0

Converting positive denary numbers to binary numbers in the two's complement format

Example 2

Convert a 38 b 125 to 8-bit binary numbers using the two's complement format.

a Since this number is positive, we must have a zero in the −128 column. It is then a simple case of putting 1-values into their correct positions to make up the value of 38:

−128	64	32	16	8	4	2	1
0	0	1	0	0	1	1	0

b Again, since this is a positive number, we must have a zero in the −128 column. As in part a, we then place 1-values in the appropriate columns to make up the value of 125:

−128	64	32	16	8	4	2	1
0	1	1	1	1	1	0	1

Writing negative binary numbers in two's complement format and converting to denary

Example 1

The following three examples show how we can write negative binary numbers in the two's complement format:

–128	64	32	16	8	4	2	1
1	0	0	1	0	0	1	1

By following our normal rules, each time a 1 appears in a column, the column value is added to the total. So, we can see that in denary this is: –128 + 16 + 2 + 1 = –109.

–128	64	32	16	8	4	2	1
1	1	1	0	0	1	0	0

Similarly, in denary this number is –128 + 64 + 32 + 4 = –28.

–128	64	32	16	8	4	2	1
1	1	1	1	0	1	0	1

This number is equivalent to –128 + 64 + 32 + 16 + 4 + 1 = –11.

Converting negative denary numbers into binary numbers in two's complement format

Consider the number +67 in 8-bit (two's complement) binary format:

–128	64	32	16	8	4	2	1
0	1	0	0	0	0	1	1

Method 1

Now let's consider the number -67. One method of finding the binary equivalent to -67 is to simply put 1s in their correct places:

–128	64	32	16	8	4	2	1
1	0	1	1	1	1	0	1

–128 + 32 + 16 + 8 + 4 + 1 = –67

Method 2

However, looking at the two binary numbers above, there is another possible way to find the binary representation of a negative denary number:

first write the number as a positive binary value – in this case 67:	0 1 0 0 0 0 1 1
we then invert each binary value, which means swap the 1s and 0s around:	1 0 1 1 1 1 0 0
then add 1 to that number:	1
this gives us the binary for –67:	1 0 1 1 1 1 0 1

1.2 Text, sound and images

1.2.1 Character sets – ASCII code and Unicode

The **ASCII code** system (American Standard Code for Information Interchange) was set up in 1963 for use in communication systems and computer systems.

The standard ASCII code **character set** consists of 7-bit codes (0 to 127 in denary or 00 to 7F in hexadecimal) that represent the letters, numbers and characters found on a standard keyboard, together with 32 control codes (that use codes 0 to 31 (denary) or 00 to 19 (hexadecimal)).

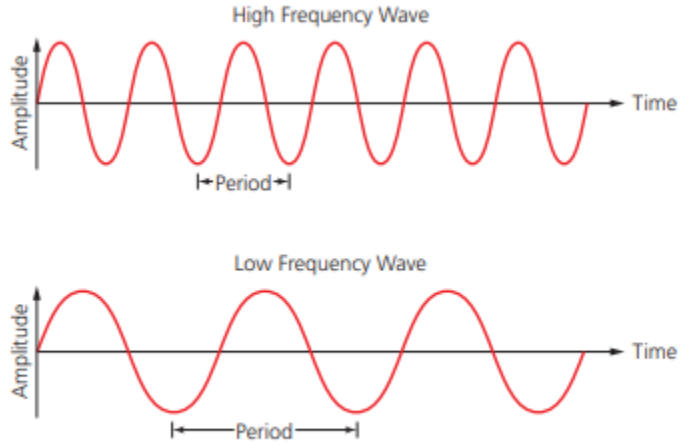
Extended ASCII uses 8-bit codes (0 to 255 in denary or 0 to FF in hexadecimal). This gives another 128 codes to allow for characters in non-English alphabets and for some graphical characters.

ASCII code lacks representation of non-Western characters, leading to **Unicode** coding system development for compatibility with DOS and Windows.

1.2.2 Representation of sound

Soundwaves are vibrations in the air. The human ear senses these vibrations and interprets them as sound.

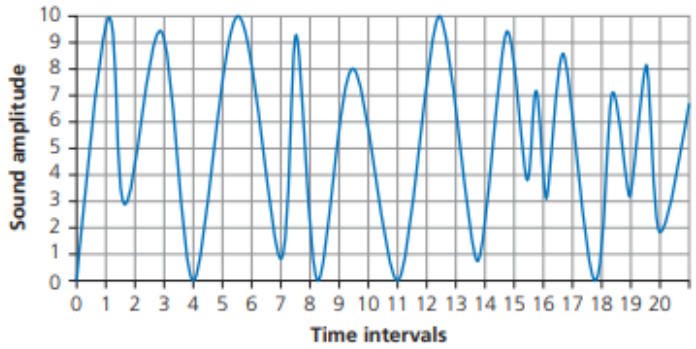
Each sound wave has a frequency, wavelength and amplitude. The amplitude specifies the loudness of the sound.



▲ **Figure 1.8** High and low frequency wave signals

To convert the analogue data to digital, the sound waves are sampled at regular time intervals. The amplitude of the sound cannot be measured precisely, so approximate values are stored.

The number of bits per sample is known as the **sampling resolution** (also known as the **bit depth**). So, in our example, the sampling resolution is 4 bits. Sampling rate is the number of **sound samples** taken per second. This is measured in hertz (Hz), where 1Hz means 'one sample per second'.



▲ **Figure 1.9** A sound wave being sampled

1.2.3 Representation of (bitmap) images

Bitmap images are made up of pixels (picture elements); an image is made up of a two-dimensional matrix of pixels. Pixels can take different shapes such as:

▲ **Figure 1.10**

The number of bits used to represent each colour is called the **colour depth**. An 8-bit colour depth means that each pixel can be one of 256 colours (because $2^8 = 256$).

Image resolution refers to the number of pixels that make up an image; for example, an image could contain 4096 × 3072 pixels (12 582 912 pixels in total).

▲ **Figure 1.11** Five images of the same car wheel using different resolutions

1.3 Data storage and file compression

1.3.1 Measurement of data storage

A **bit** is the basic unit of all computing memory storage terms and is either 1 or 0. The word comes from **b**inary dig**it**. The byte is the smallest unit of memory in a computer. 1 byte is 8 bits. A 4-bit number is called a nibble – half a byte.

▼ **Table 1.4** Memory size using denary values

Name of memory size	Equivalent denary value
1 kilobyte (1 KB)	1 000 bytes
1 megabyte (1 MB)	1 000 000 bytes
1 gigabyte (1 GB)	1 000 000 000 bytes
1 terabyte (1 TB)	1 000 000 000 000 bytes
1 petabyte (1 PB)	1 000 000 000 000 000 bytes
1 exabyte (1 EB)	1 000 000 000 000 000 000 bytes

However, since memory size is measured in terms of powers of 2, another system has been adopted by the IEC (International Electrotechnical Commission) that is based on the binary system (Table 1.5):

This system is more accurate. Internal memories (such as RAM and ROM) should be measured using the IEC system. A 64GiB RAM could, therefore, store 64 × 230 bytes of data (68719 476736 bytes).

▼ **Table 1.5** IEC memory size system

Name of memory size	Number of bytes	Equivalent denary value
1 kibibyte (1 KiB)	2^{10}	1 024 bytes
1 mebibyte (1 MiB)	2^{20}	1 048 576 bytes
1 gibibyte (1 GiB)	2^{30}	1 073 741 824 bytes
1 tebibyte (1 TiB)	2^{40}	1 099 511 627 776 bytes
1 pebibyte (1 PiB)	2^{50}	1 125 899 906 842 624 bytes
1 exbibyte (1 EiB)	2^{60}	1 152 921 504 606 846 976 bytes

1.3.2 Calculation of file size

In this section we will look at the calculation of the file size required to hold a bitmap image and a sound sample. The file size of an image is calculated as:

image resolution (in pixels) × colour depth (in bits)

The size of a mono sound file is calculated as:

sample rate (in Hz) × sample resolution (in bits) × length of sample (in seconds)

For a stereo sound file, you would then multiply the result by two.

1.3.3 Data compression

The calculations in Section 1.3.2 show that sound and image files can be very large. It is therefore necessary to reduce (or **compress**) the size of a file for the following reasons:
» To save storage space on devices such as the hard disk drive/solid state drive
» To reduce the time taken to stream a music or video file
» To reduce the time taken to upload, download or transfer a file across a network
» The download/upload process uses up network **bandwidth**
» Reduced file size also reduces costs.

1.3.4 Lossy and lossless file compression

File compression can either be lossless or lossy.

Lossy file compression

With this technique, the file compression algorithm eliminates unnecessary data from the file. This means the original file cannot be reconstructed once it has been compressed.

Common lossy file compression algorithms are:
» MPEG-3 (MP3) and MPEG-4 (MP4)
» JPEG.

MPEG-3 (MP3) and MPEG-4 (MP4)

MP3 files are used for playing music on computers or mobile phones. This compression technology will reduce the size of a normal music file by about 90%. While MP3 music files can never match the sound quality found on a DVD or CD, the quality is satisfactory for most general purposes.

MP4 files are slightly different to MP3 files. This format allows the storage of multimedia files rather than just sound – music, videos, photos and animation can all be stored in the MP4 format.

JPEG

When a camera takes a photograph, it produces a raw bitmap file which can be very large in size. These files are temporary in nature. JPEG is a lossy file compression algorithm used for bitmap images.
The JPEG file reduction process is based on two key concepts:
» Human eyes don't detect differences in colour shades quite as well as they detect differences in image brightness
» By separating pixel colour from brightness, images can be split into 8 × 8 pixel blocks,

Lossless file compression

With this technique, all the data from the original uncompressed file can be reconstructed. This is particularly important for files where any loss of data would be disastrous.

Run-length encoding (RLE) can be used for lossless compression of a number of different file formats:
» It is a form of lossless/reversible file compression
» It reduces the size of a string of adjacent, identical data (e.g. repeated colours in an image)
» A repeating string is encoded into two values:
– the first value represents the number of identical data items (e.g. characters) in the run
– the second value represents the code of the data item (such as ASCII code if it is a keyboard character)
» RLE is only effective where there is a long run of repeated units/bits.

Using RLE on text data

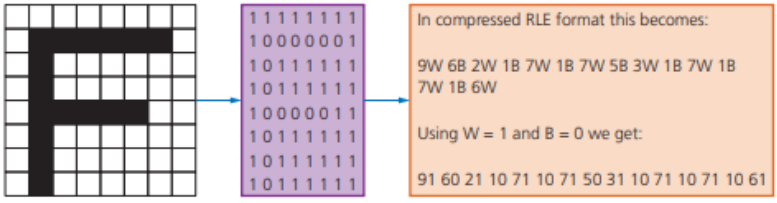
Consider the following text string: 'aaaaabbbbccddddd'. Assuming each character requires 1byte then this string needs 16bytes. If we assume ASCII code is being used, then the string can be coded as follows:

a	a	a	a	A	b	b	b	b	c	c	d	d	d	d	d
05 97					04 98				02 99		05 100				

Using RLE with images

Example 1: Black and white image

Figure 1.12 shows the letter 'F' in a grid where each square requires 1 byte of storage. A white square has a value 1 and a black square a value of 0:

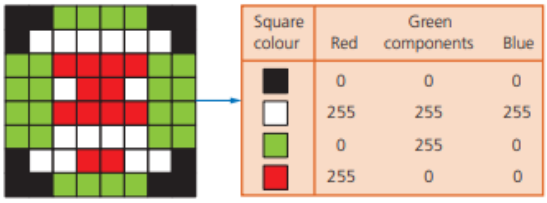


▲ **Figure 1.12** Using RLE with a black and white image

The 8 × 8 grid would need 64 bytes; the compressed RLE format has 30 values, and therefore needs only 30 bytes to store the image.

Example 2: Coloured images

Figure 1.13 shows an object in four colours. Each colour is made up of red, green and blue (RGB) according to the code on the right.



Square colour	Red	Green components	Blue
	0	0	0
	255	255	255
	0	255	0
	255	0	0

▲ **Figure 1.13** Using RLE with a coloured image

The original image (8 × 8 square) would need 3 bytes per square (to include all three RGB values). Therefore, the uncompressed file for this image is 8 × 8 × 3 = 192 bytes. The RLE code has 92 values, which means the compressed file will be 92bytes in size. This gives a file reduction of about 52%. It should be noted that the file reductions in reality will not be as large as this due to other data which needs to be stored with the compressed file

Key terms used throughout this chapter

bit – the basic computing element that is either 0 or 1, and is formed from the words **Bi**nary digi**t**

binary number system – a number system based on 2 and can only use the values 0 and 1

hexadecimal number system – a number system based on the value 16 which uses denary digits 0 to 9 and letters A to F

error code – an error message generated by the computer

MAC address – standing for Media Access Control, this address (given in hexadecimal) uniquely identifies a device on the internet; it takes the form: NN-NN-NN-DD-DD-DD, where NN-NN-NN is the manufacturer code and DD-DD-DD is the device code NN-NN-NN-DD-DD-DD

IP address – ***Internet Protocol*** identified either as IPv4 or IPv6; it gives a unique address to each device connected to a network identifying their location

HTML – ***HyperText Mark-up Language*** is used in the design of web pages and to write, for example, http(s) protocols; in the context of this chapter, colours used in web pages are assigned a hexadecimal code based on red, green and blue colours

overflow error – the result of carrying out a calculation that produces a value that is too large for the computer's allocated word size (8-bit, 16-bit, 32-bit, and so on)

logical shift – an operation that shifts bits to the left or right in a register; any bits shifted out of a register (left or right) are replaced with zeroes

two's complement – a method of representing negative numbers in binary; when applied to an 8-bit system, the left-most bit (most significant bit) is given the value –128

ASCII code – *a character set* for all the characters on a standard keyboard and control codes

character set – a list of characters that have been defined by computer hardware and software. The character set is necessary so that the computer can understand human characters

Unicode – *a character set* which represents all the languages of the world (the first 128 characters are the same as ASCII code)

sampling resolution – the number of bits used to represent sound amplitude in digital sound recording (also known as bit depth)

bit depth – the number of bits used to represent the smallest unit in a sound file

colour depth – the number of bits used to represent the colours of a pixel

sampling rate – the number of sound samples taken per second in digital sound recording

bitmap image – an image made up of pixels

pixel – derived from the term 'picture element', this is the smallest element used to make up an image on a display

image resolution – the number of pixels in the X–Y direction of an image, for example, 4096 × 3192 pixels

pixelated (image) – this is the result of zooming into a bitmap image; on zooming out the pixel density can be diminished to such a degree that the actual pixels themselves can be seen

pixel density – number of pixels per square inch

compression – reduction of the size of a file by removing repeated or redundant pieces of data; this can be lossy or lossless

bandwidth – the maximum rate of transfer of data across a network, measured in kilobits per second (kbps) or megabits per second (Mbps)

lossy (file compression) – a file compression method in which parts of the original file cannot be recovered during the decompression process for example, JPEG, mp3

lossless (file compression) – a file compression method that allows the original file to be fully restored during the decompression process, for example, run length encoding (RLE)

audio compression – a method used to reduce the size of a sound file using perceptual music shaping

MP3 – a lossy file compression method used for music files

MP4 – a lossy file compression method used for multimedia files

JPEG – from Joint Photographic Expert Group; a form of lossy file compression used with image files which relies on the inability of the human eye to distinguish certain colour changes and hues

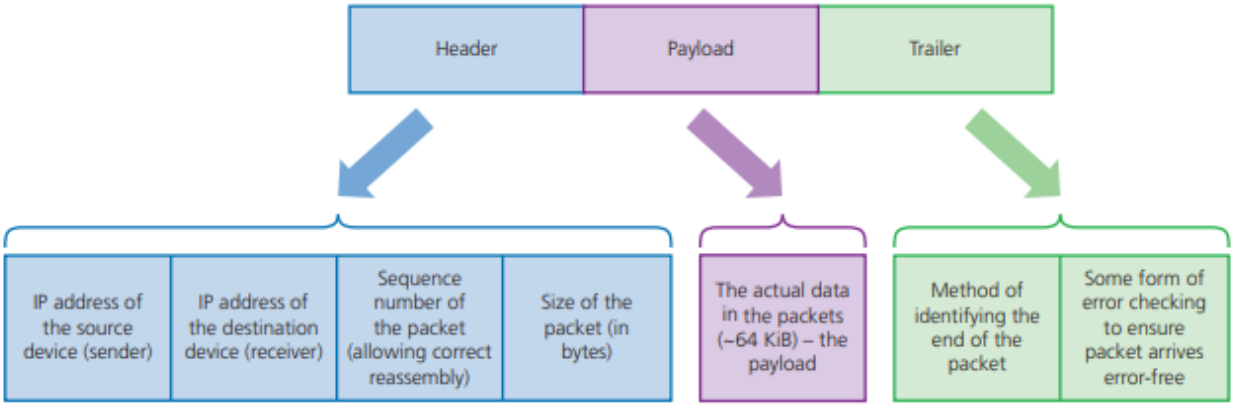
run length encoding (RLE) – a lossless file compression technique used to reduce the size of text and photo files in particular

Chapter 02 – Data transmission

2.1 Types and methods of data transmission

2.1.1 Data packets

Data sent over long distances is usually broken up into **data packets** (sometimes called **datagrams**). The only obvious drawback of splitting data into packets is the need to reassemble the data when it reaches its destination.
Packet structure A typical packet is split up into:
» A packet header
» The payload
» A trailer.



For each packet, the packet header consists of:
» The IP address of the sending device
» The IP address of the receiving device
» The sequence number of the packet (this is to ensure that all the packets can be reassembled into the correct order once they reach the destination)
» Packet size (this is to ensure the receiving station can check if all of the packets have arrived intact).

For each packet, the **payload** consists of the actual data being sent in the packet (this is usually about 64KiB).
For each packet, the **packet trailer** consists of:

» Some way of identifying the end of the packet; this is essential to allow each packet to be separated from each other as they travel from sending to receiving station
» An error checking method; cyclic redundancy checks (CRCs) are used to check data packets:
– this involves the sending computer adding up all the 1-bits in the payload and storing this as a hex value in the trailer before it is sent
– once the packet arrives, the receiving computer recalculates the number of 1-bits in the payload
– the computer then checks this value against the one sent in the trailer
– if the two values matches, then no transmission errors have occurred; otherwise the packet needs to be re-sent.

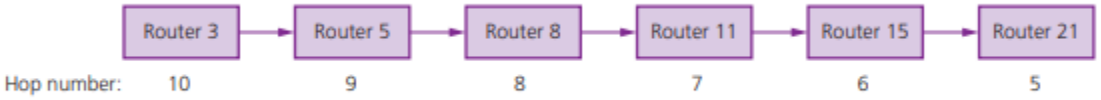
Packet switching

There will be several possible routes for the packets, between computer ‘A’ (sender) and computer ‘B’ (receiver). Each stage in the route contains a **router**

Packet switching is a method of data transmission in which a message is broken up into a number of packets. **Each** packet can then be sent independently from start point to end point.
The benefits of packet switching are:
» There is no need to tie up a single communication line
» It is possible to overcome failed, busy or faulty lines by simply re-routing packets
» It is relatively easy to expand package usage
» A high data transmission rate is possible.

The drawbacks of packet switching include:
» Packets can be lost and need to be re-sent
» The method is more prone to errors with **real-time streaming** (for example, a live sporting event being transmitted over the internet)
» There is a delay at the destination whilst the packets are being re-ordered.

Eventually the network would just grind to a halt as the number of **lost packets** mount up, clogging up the system. To overcome this, a method called **hopping** is used. A **hop number** is added to the header of each packet, and this number is reduced by 1 every time it leaves a router (Figure 2.6).



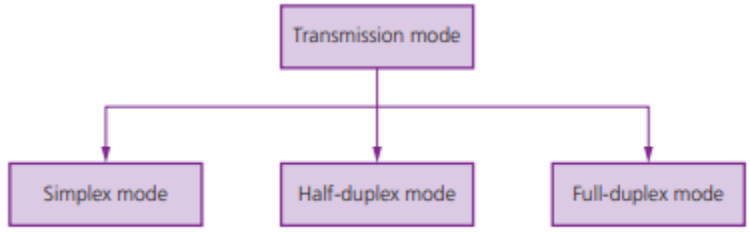
▲ **Figure 2.6** Hop numbers between routers

2.1.2 Data transmission

Data transmission can be either over a short distance

Essentially, three factors need to be considered when transmitting data:
» The direction of data transmission (for example, can data transmit in one direction only, or in both directions)
» The method of transmission (for example, how many bits can be sent at the same time)
» How will data be synch

Simplex, half-duplex and full duplex.



▲ **Figure 2.8** Transmission modes

Simplex data transmission

Simplex mode occurs when data can be sent in ONE DIRECTION ONLY (for example, from sender to receiver). An example of this would be sending data from a computer to a printer.

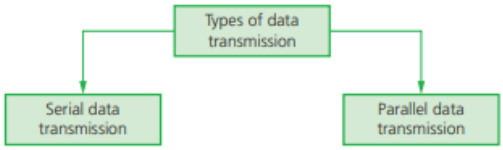
Half-duplex data transmission

Half-duplex mode occurs when data is sent in BOTH DIRECTIONS but NOT AT THE SAME TIME

Full-duplex data transmission

Full-duplex mode occurs when data can be sent in BOTH DIRECTIONS AT THE SAME TIME
Serial and parallel data transmission

Serial and parallel data transmission

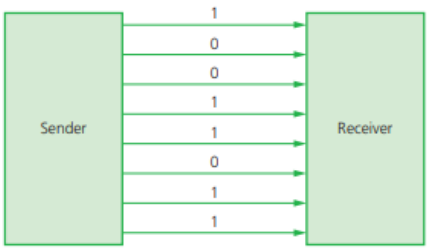


▲ **Figure 2.9** Types of data transmission

Serial data transmission occurs when data is sent **ONE BIT AT A TIME** over a **SINGLE WIRE/CHANNEL**. Bits are sent one after the other as a single stream.

▲ **Figure 2.10** Serial data transmission

Parallel data transmission occurs when SEVERAL BITS OF DATA (usually one byte) are sent down SEVERAL CHANNELS/WIRES all at the same time. Each channel/wire transmits one bit:



▲ **Figure 2.11** Parallel data transmission

Parallel data transmission works well over short distances. Over longer distances (for example, over 20metres), data can become **skewed** (that is, the data can arrive unsynchronised) and bits can arrive out of order.

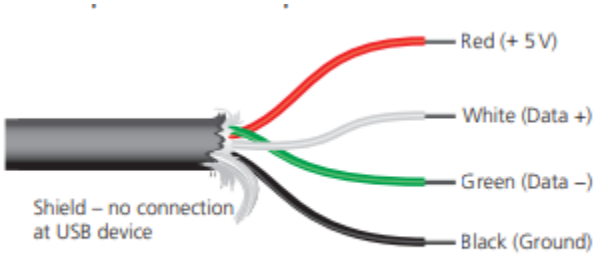
Table 2.1 shows the comparison between serial and parallel data transmission.

▼ **Table 2.1** Comparison of serial and parallel data transmission methods

Serial	Parallel
less risk of external interference than with parallel (due to fewer wires)	faster rate of data transmission than serial
more reliable transmission over longer distances	works well over shorter distances (for example, used in internal pathways on computer circuit boards)
transmitted bits won't have the risk of being skewed (that is, out of synchronisation)	since several channels/wires used to transmit data, the bits can arrive out of synchronisation (skewed)
used if the amount of data being sent is relatively small since transmission rate is slower than parallel (for example, USB uses this method of data transmission)	preferred method when speed is important
used to send data over long distances (for example, telephone lines)	if data is time-sensitive, parallel is the most appropriate transmission method
less expensive than parallel due to fewer hardware requirements	parallel ports require more hardware, making them more expensive to implement than serial ports
	easier to program input/output operations when parallel used

2.1.3 Universal serial bus (USB)

As the name suggests, the **universal serial bus (USB)** is a form of serial data transmission. USB is now the most common type of input/output port found on computers and has led to a standardisation method for the transfer of data between devices and a computer.



▲ **Figure 2.12** Typical USB cable

When a device is plugged into a computer using one of the USB ports:
» The computer automatically detects that a device is present
» The device is automatically recognised, and the appropriate device driver software is loaded up so that the computer and device can communicate effectively
» If a new device is detected, the computer will look for the device driver that matches the device; if this is not available, the user is prompted to download the appropriate driver software.

We will now consider the benefits and drawbacks of using the USB system:

▼ **Table 2.2** Benefits and drawbacks of USB systems

Benefits	Drawbacks
devices plugged into the computer are automatically detected and device drivers are automatically loaded up	standard USB only supports a maximum cable length of 5 m; beyond that, USB hubs are needed to extend the cable length
connections can only fit one way preventing incorrect connections being made	
it has become an industry standard, which means considerable support is available	even though USB is backward compatible, very early USB standards (V1) may not always be supported by the latest computers
can support different data transmission rates (from 1.5 Mbps to 5 Gbps)	
no need for external power source since cable supplies +5 V power	even the latest version 3 (V3) and version 4 (V4) USB-C systems have a data transfer rate which is slow compared to, for example, Ethernet connections (Note: USB V2 has a maximum data transfer rate of 480 Mbps.)
USB protocol notifies the transmitter to re- transmit data if any errors are detected; this leads to error-free data transmission	
it is relatively easy to add more USB ports if necessary, by using USB hubs	
USB is backward compatible (that is, older versions are still supported)	

2.2 Methods of error detection

2.2.1 The need to check for errors

When data is transmitted, there is always a risk that it may be corrupted, lost or even gained. Errors can occur during data transmission due to:
» Interference (all types of cable can suffer from electrical interference, which can cause data to be corrupted or even lost)
» Problems during packet switching (this can lead to data loss – or it is even possible to gain data!)
» Skewing of data (this occurs during parallel data transmission and can cause data corruption if the bits arrive out of synchronisation)

There are a number of ways data can be checked for errors following transmission:
» Parity checks
» Checksum
» Echo check

The parity can be either called **EVEN** (that is, an even number of 1-bits in the byte) or **ODD** (that is, an odd number of 1-bits in the byte). One of the bits in the byte (usually the most significant bit or left-most bit) is reserved for a parity **bit.**

Checksum

A checksum is a method used to check if data has been changed or corrupted following data transmission. Data is sent in blocks, and an additional value, called the checksum, is sent at the end of the block of data.

The checksum process is as follows:
» When a block of data is about to be transmitted, the checksum is calculated from the block of data
» The calculation is done using an agreed algorithm (this algorithm has been agreed by sender and receiver)
» The checksum is then transmitted with the block of data
» At the receiving end, the checksum is recalculated by the computer using the block of data (the agreed algorithm is used to find the checksum)
» The re-calculated checksum is then compared to the checksum sent with the data block
» If the two checksums are the same, then no transmission errors have occurred; otherwise, a request is made to re-send the block of data.

Echo check

With **echo check**, when data is sent to another device, this data is sent back again to the sender. The sender's computer compares the two sets of data to check if any errors occurred during the transmission process.

2.2.3Check digits

A check digit is the final digit included in a code; it is calculated from all the other digits in the code.
Check digits are used to identify errors in data entry caused by mis-typing or mis-scanning a barcode.

They can usually detect the following types of error:
» An incorrect digit entered, for example 5327 entered instead of 5307
» Transposition errors where two numbers have changed order, for example 5037 instead of 5307
» Omitted or extra digits, for example 537 instead of 5307 or 53107 instead of 5307
» Phonetic errors, for example 13 (thirteen), instead of 30 (thirty)

There are a number of different methods used to generate a check digit. Two common methods will be considered here:
» ISBN 13
» Modulo-11

2.2.4Automatic Repeat Requests (ARQs)

We have already considered parity checks and echo checks as methods to verify that data has arrived at its destination unchanged. An **Automatic Repeat Request (ARQ**) is a third way used to check data following data transmission. This method can best be summarised as follows.

» ARQ uses positive and negative acknowledgements (messages sent to the receiver indicating that data has/has not been received correctly) and **timeout** (this is the time interval allowed to elapse before an **acknowledgement** is received)
» The receiving device receives an error detection code as part of the data transmission
» If no error is detected, a positive acknowledgement is sent back to the sending device
» However, if an error is detected, the receiving device now sends a negative acknowledgement to the sending device and requests re-transmission of the data
» A time-out is used by the sending device by waiting a pre-determined amount of time ….
» ... and if no acknowledgement of any type has been received by the sending device within this time limit, it automatically re-sends the data until a positive acknowledgement is received ….
» ... or until a pre-determined number of re-transmissions has taken place
» ARQ is often used by mobile phone networks to guarantee data integrity.

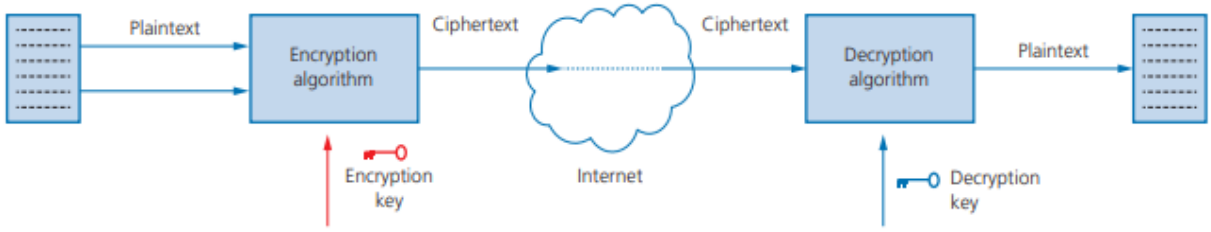
2.3 Symmetric and asymmetric encryption

2.3.1 The purpose of encryption

When data is transmitted over any public network (wired or wireless), there is always a risk of it being intercepted by, for example, a hacker. Under these circumstances, a hacker is often referred to as an **eavesdropper**. Using **encryption** helps to minimise this risk.

Plaintext and ciphertext

The original data being sent is known as **plaintext**. Once it has gone through an **encryption algorithm**, it produces **ciphertext**:



▲ **Figure 2.18** Plaintext and ciphertext

2.3.2Symmetric and asymmetric encryption

Symmetric encryption

Symmetric encryption uses an encryption key; the same key is used to encrypt and decrypt the encoded message.

However, modern computers could 'crack' this encryption key in a matter of seconds. To try to combat this, we now use 256-bit binary encryption keys that give 2256 (approximately, 1.2 × 1077) possible combinations. (Even this may not be enough as we head towards **quantum computers**.)

Asymmetric encryption

Asymmetric encryption was developed to overcome the security problems associated with symmetric encryption. It makes use of two keys called the public key and the private key:
» Public key (made available to everybody)
» Private key (only known to the computer user).

Key terms used throughout this chapter

data packet – a small part of a message/data that is transmitted over a network; after transmission all the data packets are reassembled to form the original message/data

packet header – the part of the data packet that contains the IP addresses of the sender and receiver, and includes the packet number which allows reassembly of the data packets

packet trailer – the part of a data packet that indicates the end of the data packet and cyclic redundancy check error check

cyclic redundancy check (CRC) – an error checking method in which all the 1-bits in the data packet payload are added and the total is stored in the packet trailer; the same calculation is repeated at the receiving station

payload – the actual data being carried in a data packet

node – stages in a network that can receive and transmit data packets; routers are nodes in communication networks

packet switching – a method of transmission in which a message is broken into many data packets which can then be sent along pathways independently of each other

router – a device that enables data packets to be moved between different networks, for example to join a LAN to a WAN

real time streaming – the transmission of data over a network for live events where the data is sent as soon as it is received or generated

hopping/hop number – a number in a data packet header used to stop data packets that never reach their destination from 'clogging up' the data paths/routes

simplex – data that can be sent on one direction only

half-duplex – data that can be sent in both directions but not at the same time

full-duplex – data that can be sent in both directions at the same time (simultaneously)

serial data transmission – sending data down one channel/wire one bit at a time

parallel data transmission – sending data down several channels/wires several bits at a time (usually 1 byte)

skewed (data) – data that arrives at the destination with the bits no longer synchronised

universal serial bus (USB) – a type of serial data transmission which has become the industry standard for connecting computers to devices via a USB port

parity check – a method used to check if data has been transferred correctly; it makes use of even parity (an even number of 1-bits) or odd parity (an odd number of 1-bits)

parity bit – a bit (either 0 or 1) added to a byte of data in the most significant bit position; this ensures that the byte follows the correct even parity or odd parity protocol

parity block – a horizontal and vertical parity check on a block of data being transmitted

parity byte – an extra byte of data sent at the end of a parity block; it is composed of the parity bits generated from a vertical parity check of the data block

checksum – a verification method used to check if data transferred has been altered or corrupted; calculated from the block of data of data being sent; the checksum value is sent after each data block

Revision questions

1). (2023 Nov paper1:2)
A car park has a payment machine that allows a customer to pay for their parking.
The cost of parking is displayed as a denary number on a screen on the payment machine.
The cost of parking is stored in two 8-bit binary registers.
For the parking cost of $10.50:

- register 1 stores the denary value 10 as binary
- register 2 stores the denary value 50 as binary.

(a) Give the parking cost that would be displayed on the payment machine when the registers store:

- register 1: 00010001
- register 2: 01000110
Parking cost displayed $_____ [2]

(b) The parking cost of $14.98 is displayed on the payment machine.
Give the 8-bit binary numbers that are stored in the registers to display the parking cost.
Register 1 .....
Register 2 .....[2]

(c) The payment machine gives the customer a ticket when they have paid their parking cost.
Each ticket has a 4-digit hexadecimal ticket number that is stored as binary.
The binary number 1010000000111101 is stored for a customer's ticket number.
Give the hexadecimal ticket number that would be displayed on this customer's ticket.
Hexadecimal ticket number ..... [4]

(d) Explain why data input into the payment machine needs to be converted to binary.[2]

2). (2023 nov: paper1 4)
Data is transmitted between a computer and a printer.

(a) The data is transmitted one bit at a time down a single wire. The computer can transmit data to the printer and the printer can transmit data to the computer, using the same connection.
Circle the two data transmission methods that will transmit data in this way.

parallel full-duplexparallel half-duplexparallel simplex
serial full-duplexserial half-duplexserial simplex [2]

(b) An odd parity check is used to detect errors in the data transmission.
Explain how the odd parity check detects errors.[4]

(c) Another error detection method sends the data from the computer to the printer, then a copy of the data received is sent back from the printer to the computer. The two sets of data are compared to see if they match.
State the name of this type of error detection method.[1]

3). [2023 Nov: paper1 5]
A musician is recording herself playing the music for a song on the piano.

(a) Explain how the analogue sound is recorded and converted to digital.[5]

(b) State two ways that the accuracy of the recording could be increased.[2]

(c) The musician compresses the sound file using lossless compression instead of lossy compression.
Explain why the musician would choose to use lossless compression instead of lossy compression.[3]
(d) The musician types the words for the song into a document.
Two-character sets that can be used when converting text to digital are the American standard code for information interchange (ASCII) and Unicode.
Explain the differences between the ASCII character set and the Unicode character set.[4]

4). [2023 Nov: paper1 Computer systems 2]
A register stores the binary number:

1	1	1	0	0	0	1	1

(a) Give the denary number for the binary number stored in the register.[1]

(b) Give the hexadecimal number for the binary number stored in the register. [2]

(c) A logical left shift of two places is performed on the binary number stored in the register.
Complete the binary register to show its contents after this logical left shift.

[1]

(d) The negative denary number −99 needs to be stored in the register.
Complete the register to show the binary number that would be stored, using two's complement. Show all your working.

[2]

(e) The number 01001100 is added to 11100011
Add the two 8-bit binary numbers, using binary addition.
Give your answer in binary. Show all your working.[4]

5). [2023 Nov: paper1 Computer systems 5]
A band is recording their new song. They need to consider the sample rate and sample resolution of their recording.

(a) Give one benefit of using a higher sample rate to record the song.[1]

(b) Give one drawback of using a higher sample rate to record the song.[1]

(c) Describe what is meant by sample resolution.[2]

(d) The band wants to compress the sound file, but they do **not** want any data to be permanently removed.
Identify the compression method that should be used. [1]

6). [2023 Nov: paper1 Computer systems 6]
The table contains descriptions about data transmission methods.
Complete the table by identifying which data transmission methods are described.

Data transmission method	Description
.....	Data is transmitted down a single wire, one bit at a time, in one direction only.
.....	Data is transmitted down multiple wires, multiple bits at a time, in both directions, but only one direction at a time.
.....	Data is transmitted down a single wire, one bit at a time, in both directions at the same time.
.....	Data is transmitted down multiple wires, multiple bits at a time, in one direction only.

[4]

7). [2023 Nov: paper1* Computer systems 2]
Humans use a denary number system, and computers use a binary number system.

(a) Explain what is meant by a binary number system. [2]

(b) Convert the denary numbers 14, 59 and 234 to binary.
14, 59, 234 [3]

(c) Convert the denary numbers 9, 26 and 65 to hexadecimal.
9, 26, 65 [3]

(d) Convert the positive denary number 123 to 8-bit binary using two's complement.
Show all your working. [2]

(e) Add the binary values 00110011 and 01111000 using binary addition.
Give your answer in binary. Show all your working. [3]

8). [2023 Nov: paper1 Computer systems 5]
5 Errors can occur when data is transmitted.

(a) Give one reason an error may occur when data is transmitted. [1]

(b) Some error detection methods use a calculated value to check for errors.
Tick (✓) one box to show which error detection method does not use a calculated value to check for errors.

- A Check digit
- B Checksum
- C Echo check
- D Parity check [1]

(c) An automatic repeat request (ARQ) can be used to make sure that data is received free of errors. It can use a positive or negative acknowledgement method to do this.
Explain how an ARQ operates using a positive acknowledgement method. [5]

9). [2023 Nov: paper1 Computer systems 7]
A photographer takes an image with a digital camera. The photographer sets the resolution and colour depth for the image.

(a) State what is meant by the image resolution.[1]

(b) State what is meant by the image colour depth.[1]

(c) Give one benefit of increasing the colour depth of the image.[1]

(d) The photographer compresses the image using a method that permanently reduces the colour depth and resolution of the image.
Identify which compression method the photographer uses.[1]

(e) One benefit for compressing the image is to reduce the storage space it uses.
Give two other benefits of compressing the image.[2]

10). [2022 May: paper2 1]
Jack has an MP3 file stored on his computer.

(a) (i) Tick (✓) to show which type of data is stored in an MP3 file.
Tick (✓)

- Video
- Sound
- Image [1]

(ii) Tick (✓) to show whether the MP3 file is a lossy compressed file or a lossless compressed file or **not** a compressed file.
Tick (✓)

- Lossy compressed file
- Lossless compressed file
- **Not** a compressed file [1]

11). [2022 May: paper2 4]
All data needs to be converted to binary data so that it can be processed by a computer.

(a) Explain why a computer can only process binary data.[2]

(b) The denary values 64, 101 and 242 are converted to 8-bit binary values.
Give the 8-bit binary value for each denary value.
64, 101, 242 [3]

(c) The hexadecimal values 42 and CE are converted to binary.
Give the binary value for each hexadecimal value.
42, CE [4]

12). [2022 May: paper2 5]
An image is stored on a computer. The image is 16-bit colour and is 100 pixels high and 150 pixels wide.
Calculate the file size of the image in bytes. Show all your working.
Answer ..... bytes [3]

13). [2021 Feb: paper1 1]
A hockey club records the number of people that watch each match. An 8-bit binary register is used to store this value.

(a) 46 people watch the first match and 171 people watch the second match.
Show how the registers would store these denary values as 8-bit binary.

Denary value	8-bit binary							
46								
171								

[2]

(b) Give the largest denary value that can be stored in the 8-bit binary register. [1]

(c) The hockey club wants to increase the number of people that can watch each match to 2000.
The 8-bit binary register may no longer be able to store the value.
Give the smallest number of bits that can be used to store the denary value 2000.[1]

(d) Electronic data about the final score for the match is transmitted to a central computer 30 kilometres away, using serial transmission.

(i)Explain why serial transmission is more appropriate than parallel transmission in this scenario.[3]

(ii)The data transmission is also half-duplex.
Describe half-duplex data transmission.[2]

(iii)The data transmission uses checksums.
Describe how checksums are used to detect errors in data transmission. [3]

13). [2020 nov: paper1 2]
Ron is attending a music concert. He has bought three tickets.
Each ticket number is displayed as a hexadecimal number.

(a) Complete the table to show the 12-bit binary values and the Denary values for each Hexadecimal ticket number.

Hexadecimal ticket number	12-bit binary value	Denary value
028		
1A9		
20C		

[6]

14). [2020 nov: paper1 7]
Nina is recording some music tracks that she has written. She is researching whether she should record them in MIDI or MP3 format.
Explain what is meant by MIDI and MP3 format. [3]

15). [2020 nov: paper1 9]
Victoria is entering data into a computer system. The data will be transmitted to cloud storage.

(a) An even parity check is used to check for errors in the binary values after transmission.
For each of the 7-bit binary values, write the Parity bit that makes sure even parity is met.
7-bit binary value Parity bit
1100010 ……

1001011 .....
0100010 .....
0010111 ..... [4]

(b) Identify **two** other error checking methods that could be used to check the binary values are correct after transmission. [2]
Method 1

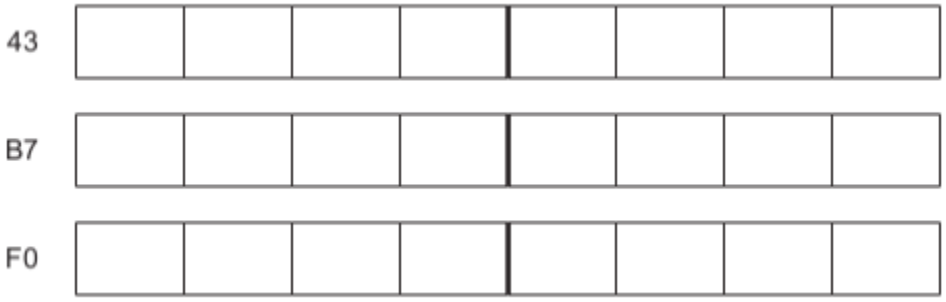
_____

Method 2

_____

(c) A check digit is used to check whether data is correct when entered into the system.
Describe how a check digit can be used to make sure the data entered is correct.[4]

16). [2020 nov: paper1* 1]
The hexadecimal colour code #43B7F0 is stored in three 8-bit registers.
Give the 8-bit binary values for each part of the hexadecimal code.



[6]